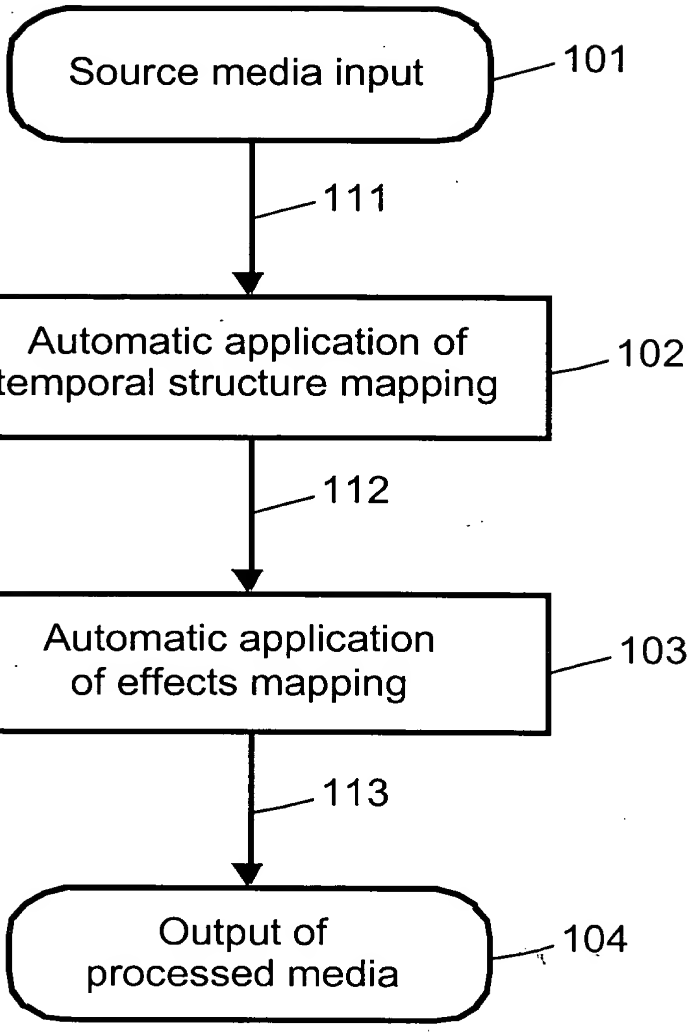
Source media input
101
111
Automatic application of temporal structure mapping
102
112
Automatic application of effects mapping
103
113
Output of processed media
104

Fig. 1

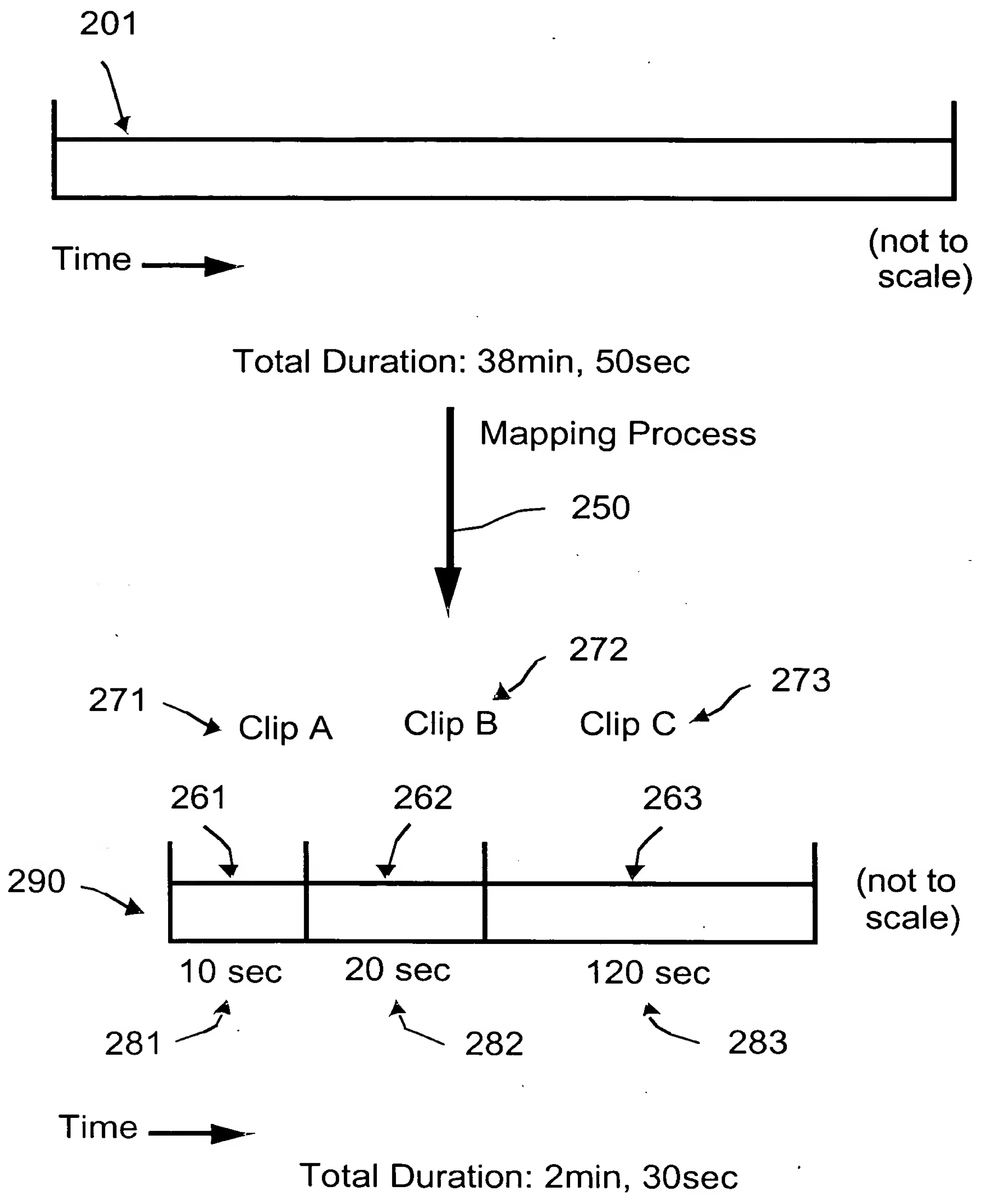
201
Time
(not to scale)
Total Duration: 38min, 50sec
Mapping Process
250
271
Clip A
272
Clip B
273
Clip C
261
262
263
290
(not to scale)
10 sec
20 sec
120 sec
281
282
283
Time
Total Duration: 2min, 30sec

Fig. 2

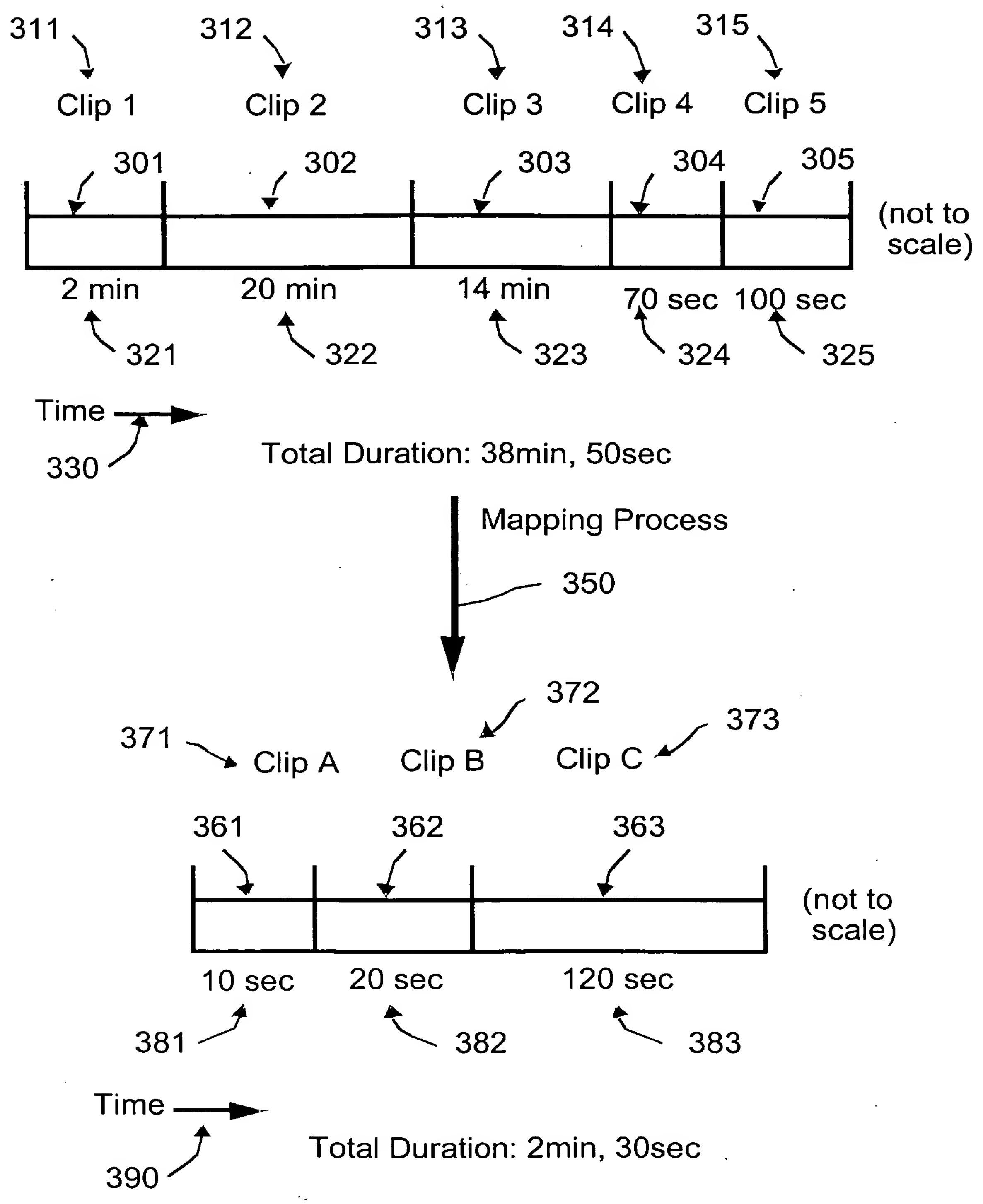
311
312
313
314
315
Clip 1
Clip 2
Clip 3
Clip 4
Clip 5
301
302
303
304
305
(not to scale)
2 min
20 min
14 min
70 sec
100 sec
321
322
323
324
325
Time
330
Total Duration: 38min, 50sec
Mapping Process
350
372
371
Clip A
Clip B
Clip C
373
361
362
363
(not to scale)
10 sec
20 sec
120 sec
381
382
383
Time
390
Total Duration: 2min, 30sec

Fig. 3

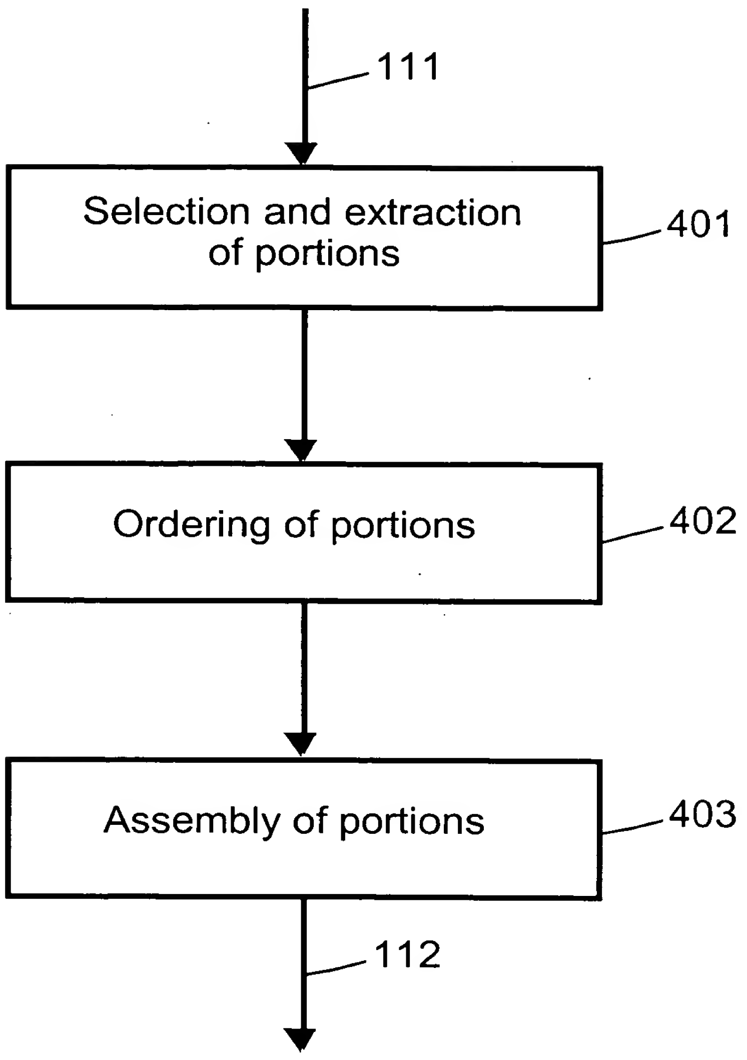
111
Selection and extraction of portions
401
Ordering of portions
402
Assembly of portions
403
112

Fig. 4

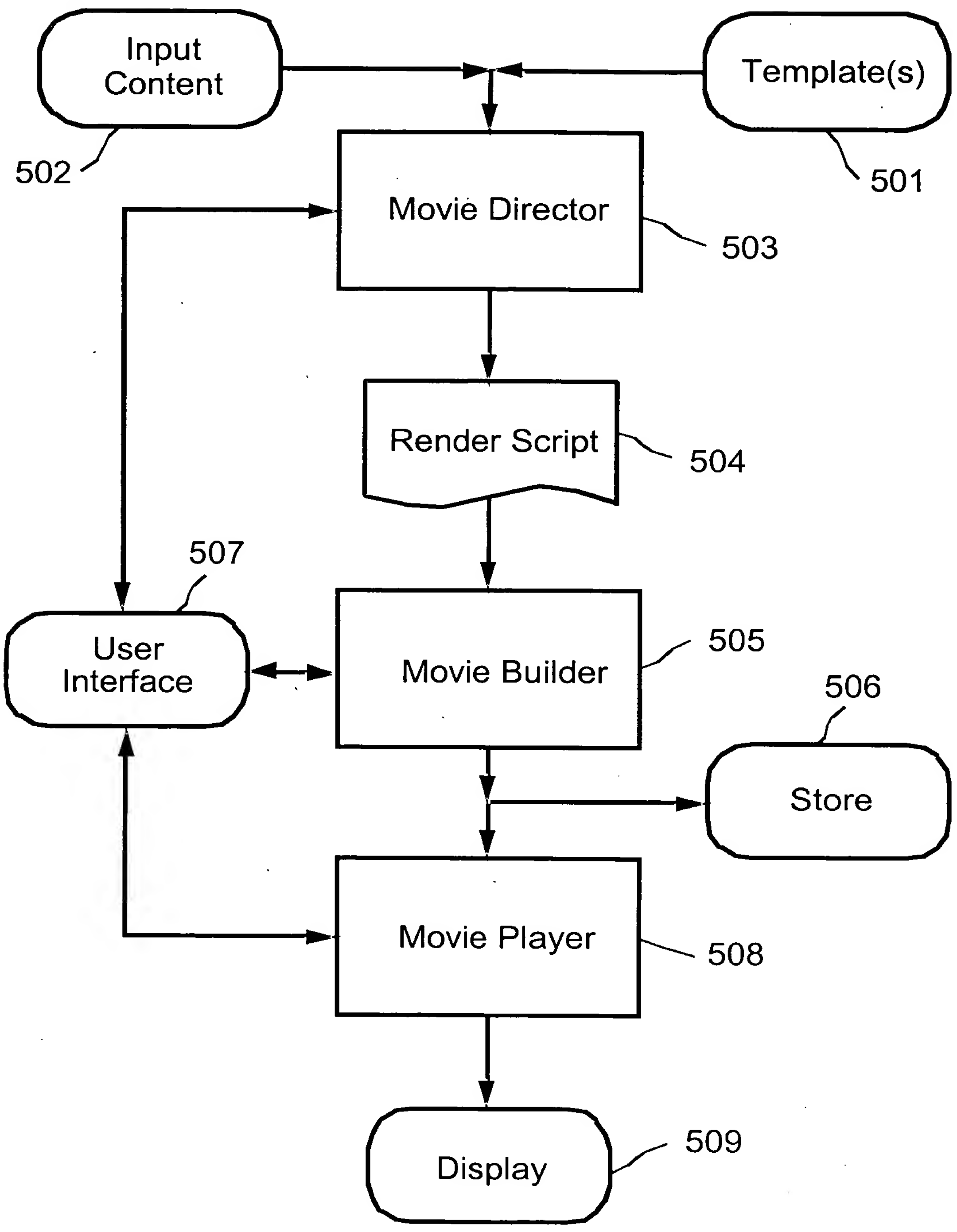
Input Content
Template(s)
502
501
Movie Director
503
Render Script
504
507
User Interface
Movie Builder
505
506
Store
Movie Player
508
Display
509

Fig. 5

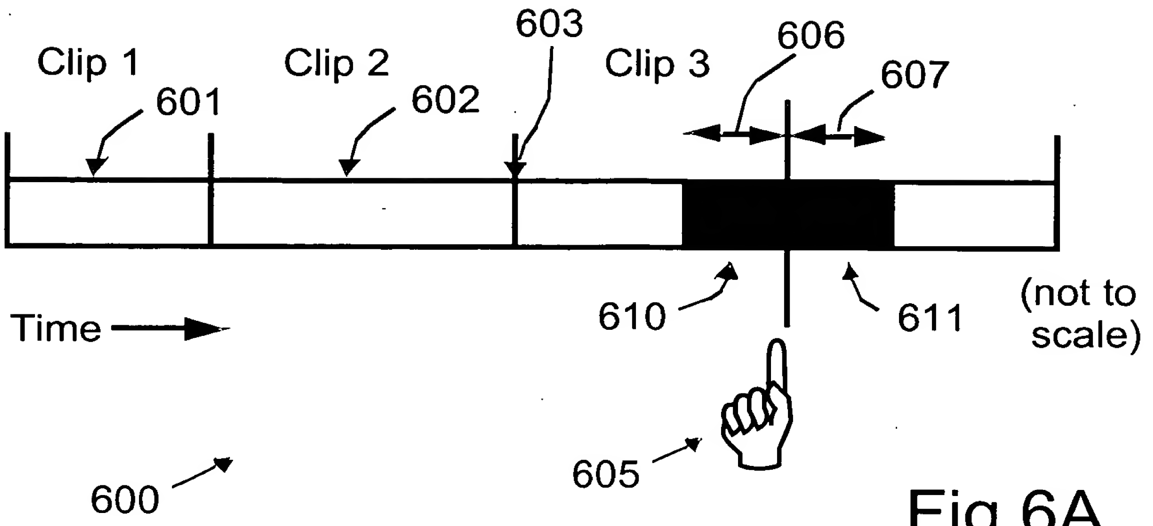
Clip 1
601
Clip 2
602
603
Clip 3
606
607
610
611
(not to scale)
Time
605
600

Fig 6A

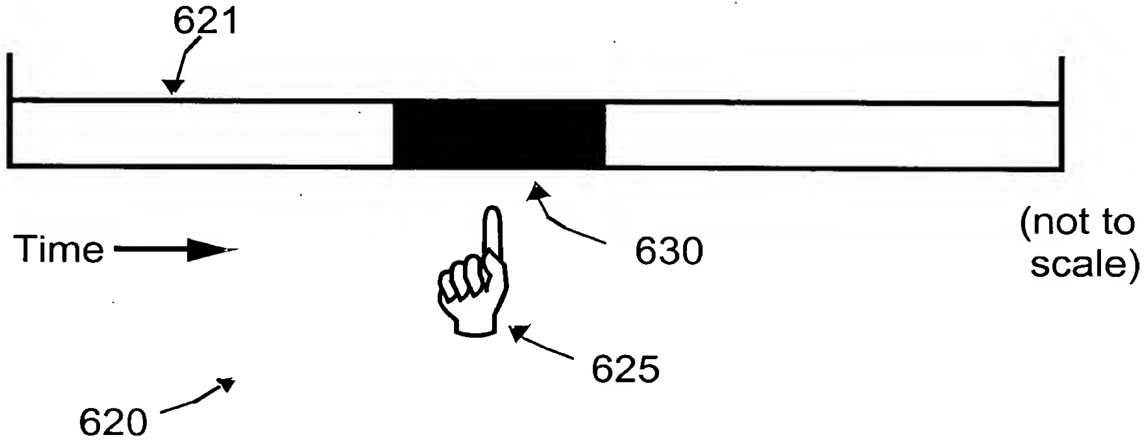
621
630
(not to scale)
Time
625
620

Fig. 6B

Table 1.

Selection and Extraction Method Examples		
Selected Portion Start	**Extracted Duration**	**Relationships between Selections**
Within whole content.	Random.	Random, chronological, without overlap.
Within clip.	Less than or equal to clip length.	Random, chronology ignored, overlap ignored.
Within a group of clips.	Spanning one or more clips recorded within the same day.	From separate clips.
Heuristically obtained, eg. assume zones of interest in recorded content occur primarily near clip startpoints.	Heuristically obtained, eg. related to human attention span.	From a group of clips recorded within the same day.
Multi-pass (repetitious)	Limited so as to limit total output duration (eg. based on heuristics).	From all clips within whole content.
	Short durations followed by longer durations (eg. applied to multipass selection)	Repetitious, for instance, to lengthen output content duration with respect input content duration.

Table 2.

Ordering Method Examples
Sequential or chronological
Random
Reverse-chronological
Flashback (later chronologies displayed or duplicated early in the order)
Montage (later chronologies displayed in brief early in the order)
Cutaway (two related or consecutive portions separated by an unrelated or distant portion)
Alternate

Table 3.

Assembly Method Examples
Cut (butt-edit)
Short Dissolve
Long Dissolve
Fast Wipe
Slow Wipe
Graphic

Table 4.

Effects Mapping Examples
Addition of Sound effect
Removal of chrominance
Addition of artificial scratches and dust
Composition or overlay of sprites, animation, graphics
Addition of Music
Luminance or chrominance keying or matteing
Dissolve or mixing of other content

Table 5

Silent Movie Template Components Example	
Component	**Purpose**
Four well-separated random video selections from input content.	Selection of sufficiently differing activities or incidents from the input content to create surprise or reduce boredom.
Extract limited duration clips for each selection, each preferably less than 2 minutes in duration.	Limit clip duration to the effective viewer attention span and avoid boredom.
Filter clips to remove all chrominance information.	Replicate "black and white" characteristic of Silent Movie genre.
Remove original audio information.	Replicate silent characteristic of Silent Movie genre.
Add piano soundtrack.	Replicate characteristic of Silent Movie genre.
Insert dialogue mattes at clip boundaries.	Replicate characteristic of Silent Movie genre.
Apply scratch and dust filter.	Replicate characteristic of Silent Movie genre.
Cut in titles, dialogue mattes and clips.	Replicate hard-cut characteristic of Silent Movie genre.
Insert fade-in from black to title dialogue matte.	Include title in characteristic style of Silent Movie genre.
Insert fade-out to black from end-title dialogue matte.	Include end-title in characteristic style of Silent Movie genre.
Insert film projector sprocket hole sound over title.	Replicate projector sound-effect characteristic of Silent Movie genre.

Table 6.

Example Associations between Editing & Effect Techniques and Template Type

	Romance Montage	Action Montage	Continuity Template	Silent Movie
Transitions				
Fade				
Fade out	✓		✓	✓
Fade in	✓		✓	✓
Dissolve	✓			
Cross-fades	✓		✓	
Wipe		✓		
Quick/Whip		✓		
Audio	✓		✓	
Sound Types				
Actual Sound	✓	✓	✓	
Sound effects	✓	✓	✓	✓
Atmos sound	✓	✓	✓	✓
Voice over	✓	✓	✓	
Cuts				
Cross cut	✓	✓		
Continuity cut	✓	✓	✓	✓
Compilation cuts		✓		
Split editing	✓		✓	
Parallel cutting				
Classical cutting	✓		✓	✓
Editing effects				
Cutaways	✓	✓	✓	
Insert	✓	✓	✓	
Subliminal cuts				
Flashbacks		✓	✓	
Freeze-frames	✓	✓		
Frequency	✓	✓		
Duration				
Montages	✓	✓		
Rhythm	✓	✓	✓	
Reverse shot	✓	✓	✓	
Shot length				
Same length			✓	
Slow cutting	✓			
Fast cutting	✓	✓		
Cut to beat/music	✓	✓		

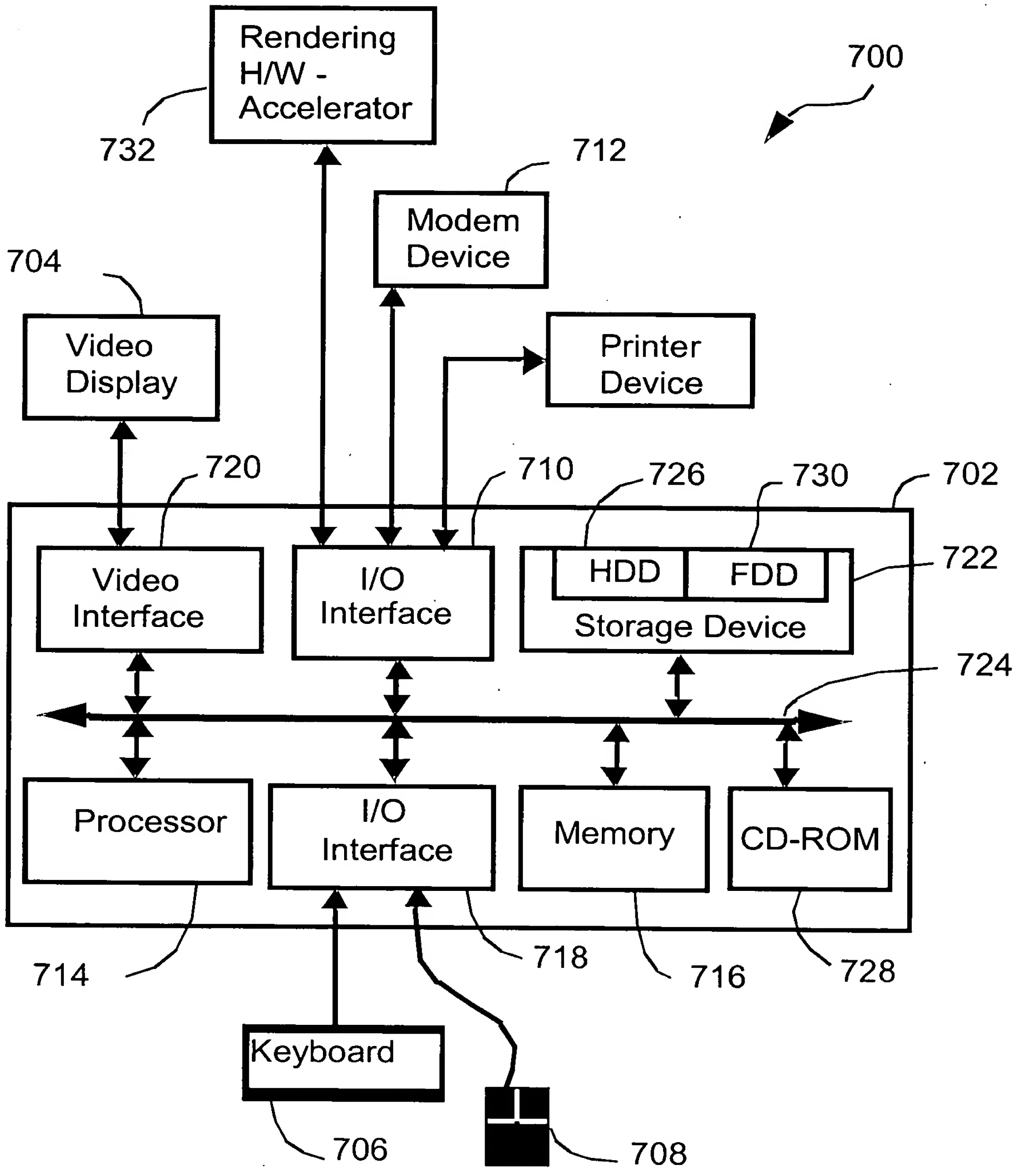
Rendering H/W - Accelerator
732
700
712
Modem Device
704
Video Display
Printer Device
720
710
726
730
702
Video Interface
I/O Interface
HDD
FDD
Storage Device
722
724
Processor
I/O Interface
Memory
CD-ROM
714
718
716
728
Keyboard
706
708

FIG. 7

Appendix 1

Movie Director Example Implementation (Pseudo-code)

```
main()
begin
        create rule list
        create parameter list
        create item list
        create rule syntax table

        get template file name
        load(template_file_name)

        get render script file name
        create render script file

        get input content file names
        create content list
        contentparse(content_list, input_content_file_names,...)

        ruleparse(installed_rules, content_list, render_script_file)
        save render_script_file
        close render_script_file
        exit
end

load(template_file_name)
begin
        while(not end of template_file)
        begin
                get next item
                if (item_type == reference)
                        resolve(item)
                else if (item_type == rule)
                        install rule_name
                else if (item_type == parameter)
                        write(parameter_list, parameter_name)
                else if (item_type == rule_syntax_extension)
                        write(rule_syntax_table, rule_syntax_extension)
                else
                        write(item_list, item_name)
        end
end

resolve(reference_name)
begin

                if (reference_type == provided_content)
```

```
                    begin
                            get provided content file name
                            contentparse(content_list, provided_content_file_name)
                    end
                    else
                            get referenced item
            end
end


contentparse(content_list, content_file_name, ...)
begin
        while(not last content item)
        begin
                if (content_file_name_type == directory)
                        begin
                        get directory contents
                        contentparse(content_list,
                                        directory_content_file_names,...)
                        end
                else
                        begin
                        get content information
                        write(content_list, content_file_name, content_information)
                        end
        end
end


ruleparse(rule_list, content_list, render_script_file_name)
begin
        create instruction list
        while (not last rule)
        begin
                get rule
                decode(instructions, operands, rule, content_references,
                        parameter_references, item_references)
        end

        get instruction list
        while (not last instruction)
        begin
                execute instruction(operands)
        end
end

decode(instructions, operands, rule, content_references, parameter_references,
        item_references)
begin
        while (not end of rule)
        begin
                get next portion
                if (portion_type == instruction)
                        begin
                        read(portion)
                        convert portion according to rule syntax table
                        write(instruction_list, instruction)
                        end
                else
                        begin
                        read(reference)
                        convert portion according to rule syntax table
                        write(instruction_list, operand)
                        end
        end
end
```

Appendix 2

Movie Builder Example Implementation (Pseudo-code)

```
main()
begin
        get render script file name
        get destination movie file name
        open render script file
        create qt_movie_file
        parse(render_script_file, qt_movie_file)
        close render_script_file
        save qt_movie_file
        close qt_movie_file
        exit
end

parse(script_file_name, qt_movie_file)
begin
        while(not end of script_file)
        begin
                get next script file line
                parse_line(script_file_line, qt_movie_file)
        end
end

parse_line(script_file_line, qt_movie_file)
begin
        get first word of line
        if "//" return
        else if "video" then
                video(script_file_line, qt_movie_file)
        else if "audio" then
                audio(script_file_line, qt_movie_file)
        else if "transition" then
                transition(script_file_line, qt_movie_file)
```

```
          else
                    flag error in script file
end

video(script_file_line, qt_movie_file)
begin
          parse video paramenters
          add video to qt_movie_file using QT API
end

audio(script_file_line, qt_movie_file)
begin
          parse audio paramenters
          add audio to qt_movie_file using QT API
end

transition(script_file_line, qt_movie_file)
begin
          parse transition paramenters
          add transition to qt_movie_file using QT API
end
```

Appendix 3

Template Example Implementation (Pseudo-code)

```
//Action Template                      Fast-paced, quick cutting, fast beat.
cut_order = chronological              //chronology not strictly enforced
structure = 10s, 4s, ...               //repetitive temporal structure
intraclip_cutting = 2                  //one long clip may contribute 2 elements
intraclip_spacing = 2s
avoid_cutting = 1s, -1s                //do not use first/last second of clip
cut_method = random, clip
play_order = forward
structure_transition = 3, 4, crossfade  //3-4 frame crossfade
beat_synchronise = true                //sync video clip lengths to music beat
back_track = action                    //specify backing music characteristics
audio_action = mute_all                //remove all original audio
title = action_title
end_title = action_end_title

//function definition

length check_fit(content-length, structure, intraclip_spacing, intraclip_cutting,
                 avoid_cutting)
begin
    length = content_length – avoid_cutting[0] + avoid_cutting[1] - structure[0]
    x = intraclip_cutting
    while (x > 1)
    begin
        x = x - 1
        length = length – structure[x] – intraclip_spacing[x]
    end
    return length
end
```

```
main()          //start
begin
        trim (title, beat_synchronise, structure_transition)
        assemble_edit (output, title, play_order, structure_transition, audio_action,
                          back_track)
        while not (completed content list)
        begin
                get next content (cut_order)
                excess = check_fit (content_length, structure, intraclip_cutting,
                          intraclip_spacing, avoid_cutting)
                if (excess > 0)
                begin
                     y = 0
                     cut_start = cut_method(excess)
                     cut_end = 0
                     while (y < intraclip_cutting)
                     begin
                          cut_end = cut (avoid_cutting + cut_start + cut_end,
                                         structure[y])
                          y = y + 1
                          cut_start = excess – cut_start
                     end
                end

                trim (current_clips, beat_synchronise, structure_transition)
                assemble_edit (output, current_clips, play_order, structure_transition,
                                  audio_action, back_track)
        end
        trim (end_title, beat_synchronise, structure_transition)
        assemble_edit (output, end_title, play_order, structure_transition, audio_action,
                          back_track)
end //finish
```